

Umstellung von FVL-Formularen für den PDF-Export auf HTML-Formulare

– Stand 16.01.2015 –

Formular vorbereiten (pro Formular)

- FVL im eforms&more Formular-Editor öffnen
- Unter Menü „Skripte“ den Punkt „**PDF-Javaskripte in HTML-Javaskripte kopieren ...**“ aufrufen und mit „Ok“ bestätigen.

Es wird angezeigt, wie viele Skripte (einschließlich des globalen Javaskripts) übernommen werden.

Hinweis: Die JavaScripte werden (fast) 1:1 nach HTML übernommen und in HTML durch Erweiterung der Script-Sprache direkt im Browser interpretiert.

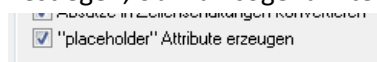
Hinweis: Bei diesem Vorgang werden auch die „Dateiformate für die Übertragung“ von PDF nach HTML übernommen. Dadurch wird sichergestellt, dass beim Absenden gleiche Formate genutzt werden.

Hinweis: PDF-JavaScripte werden nur kopiert, wenn es noch keine entsprechenden HTML-JavaScripte gibt. Daher kann an den HTML-JavaScripten im Nachhinein korrigiert werden, ohne dass bei einem erneuten Kopieren die HTML-Skripte überschrieben würden.

- Unter Menü „Formular“ den Punkt „**Einstellungen für den Export nach HTML ...**“ aufrufen.

Unter dem Reiter „**Allgemein**“:

Festlegen, ob man sogenannte „placeholder“ verwenden möchte.



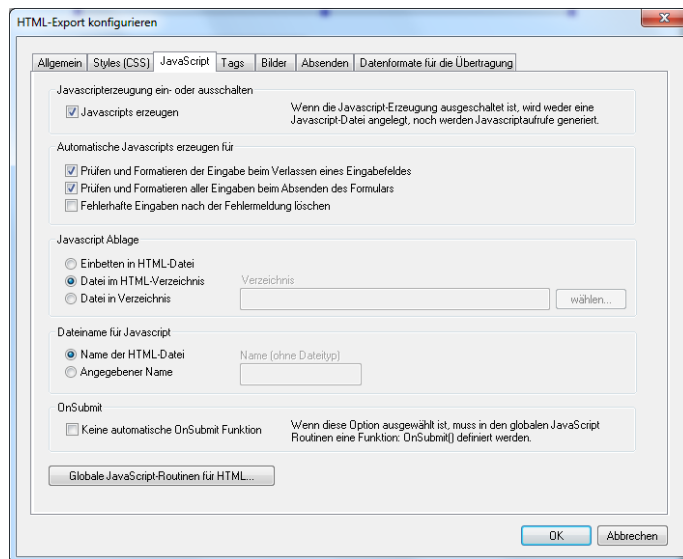
Bitte Geburtsort eingeben

Wenn dies angekreuzt ist, dann wird der „Text in der Statuszeile“ in alle modernen Browsern als Hintergrund eingeblendet, solange das Feld noch nicht gefüllt wurde.

Vorteil: Es unterstützt den Nutzer beim Ausfüllen.

Nachteil: der Text ist oft zu lange für das Feld und wird abgeschnitten. Der Text kommt außerdem bei allen Browsern immer als „Mouse-Over“ Text.

Unter dem Reiter „JavaScript“:

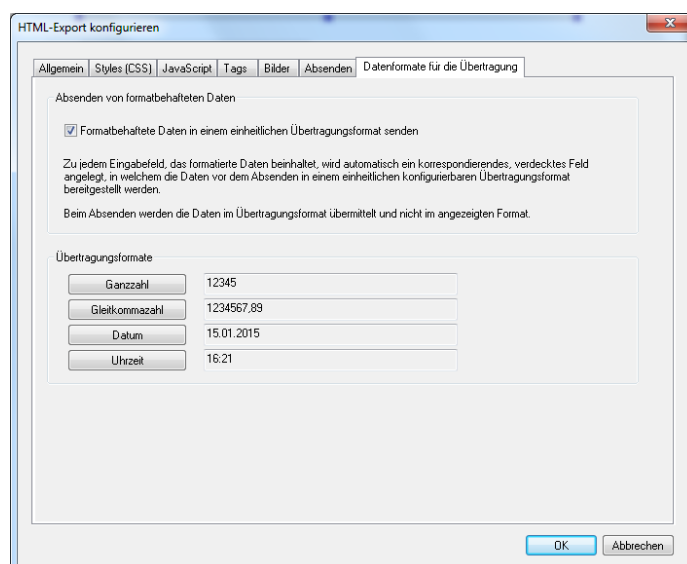


Das Häkchen bei „Prüfen und Formatieren der Eingabe beim Verlassen ...“ sollte gesetzt sein. Das Häkchen bei „Fehlerhafte Eingaben nach Fehlermeldung löschen“ sollte gesetzt werden, wenn das gleiche Verhalten wie im PDF gewünscht wird oder wenn das Formular nicht einreichbar ist, andernfalls sollte „Prüfen und Formatieren ... beim Absenden ...“ angekreuzt werden.

D.h. immer das ersten und eines der beiden weiteren Kästchen bei „Automatisches JS“ ankreuzen.

Ob man das JavaScript als eigene Datei erzeugen lässt oder in den HTML-Code einbettet ist Geschmackssache. Einstellung unter „Javascript Ablage“.

Unter dem Reiter „Dateiformate für die Übertragung“:



die gewünschten Formate für die Übermittlung an das Gateway (-> XML) festlegen oder oben das Häkchen entfernen.

Hinweis: Beim Kopieren der JavaScripte wurden auch die „Dateiformate für die Übertragung“ von PDF nach HTML übernommen und sollte daher passen, wenn Sie ein früheres PDF-Formular umstellen.

Unter dem Reiter „Absenden“:

The screenshot shows the 'HTML-Export konfigurieren' dialog box with the 'Absenden' tab selected. The 'Frei definierte Aktion' radio button is selected. The 'Methode' dropdown is set to 'post' and the 'Codierung' dropdown is set to 'multipart/form-data'. There are also fields for 'Email-Adresse', 'Betreff', and 'Aktion'.

„Frei definierte Aktion“ auswählen und als Methode „post“ auswählen.

Nur wenn das Formular auch die Möglichkeit von **Dateianlagen** enthält, dann bitte bei Codierung auf „multipart/form-data“ umstellen.


Das Feld „Aktion“ kann leer bleiben. Diese Werte werden bei einem „submit“ verwendet. Die Zieladresse wird – wie bisher - aus dem unsichtbaren Feld „TargetURL“ ausgelesen.

- Prüfen, ob das Formular **„Dateianlagen“** enthält.
Wenn ja, die Buttons für den Upload löschen/ausblenden (werden in HTML direkt vom jeweiligen Browser – im eigenen Format – erzeugt) und das Feld für den Dateianhang auf „veränderbar“ abändern.
Außerdem die Codierung für das Absenden entsprechend einstellen (siehe oben).
- Prüfen, ob im Formular **mit Zahlen gerechnet** wird (Evtl. STRG-Shift-Q verwenden).
Wenn ja muss (vor allem bei Additionen) statt **xxx.value** nun **xxx.valueAsInt** bzw. **xxx.valueAsFloat** verwendet werden. Die Funktion AFMakeNumber für die Umwandlung in Gleitkommazahlen im PDF-JavaScript kann für HTML entfallen, wenn **xxx.valueAsFloat verwendet wird**.
- Prüfen, ob im Formular mit **Datumsangaben gerechnet** wird.
Wenn ja, die Nutzung der Funktionen „util.formatd“ und „util.scand“ sehr gut kontrollieren (bisher nur für Standard-Fälle umgesetzt). Evtl. diese Berechnungen direkt als HTML-JS umsetzen.
- Wenn im JavaScript mit **„event.value“** gearbeitet wird, muss dieses in entsprechendes HTML-JavaScript umgewandelt werden.
- Wenn im Formular **längere mehrzeilige RTF-Texte** verwendet werden, kann man ebenfalls im Menü „Formular“ unter dem Punkt „Einstellungen für den Export nach HTML ...“ -> „Allgemein“ festlegen, ob Absätze in Zeilenschaltungen konvertiert werden soll, da die Texte in manchen Browsern (z.B. Firefox) sonst deutlich zu lang werden.
 Absätze in Zeilenschaltungen konvertieren
- **Hinweise** was in HTML nicht geht bzw. gehen kann:
 - **Senkrechte Texte** oder **senkrechte Felder**(z.B. am linken Seitenrand)
Entweder nur nach PDF exportieren oder anderswo platzieren
 - Alle **schrägen Striche** die nicht senkrecht oder waagrecht sind
Kommen sehr selten vor. Im Bedarfsfall evtl. ein Bild verwenden.
 - **Eckige Ankreuz-Gruppen** (Radio-Buttons) werden in HTML vom Browser immer rund dargestellt.
Bei Bedarf Rechtecke im Hintergrund und Radio-Button ohne Rand im Vordergrund nutzen.

- **Größe von Checkboxes und Radiobuttons** wird im HTML immer gleich groß (Browser-Standard). Position der Ankreuzfelder evtl. über „Einstellungen für den Export nach HTML ...“ -> „Allgemein“ leicht nach links und oben verschieben (z.B. 1 mm). Das Layout sieht in HTML dann in manchen Browsern besser aus. *Wenn wirklich größere Ankreuzfelder nötig sind, dann evtl. auf Textfelder mit individuellen „onClick“ Events wechseln.*
- Positionskorrekturen für Ankreuzfelder

Nach links	Nach oben
0,0 mm	0,0 mm
- **Transparente Eingabefelder** gibt es in HTML nicht. Linien/Tabellenränder werden von Feldern überdeckt. *Lösung: sehr genau positionieren.*
 - **Kammfelder** (z.B. für BLZ) gibt es in HTML nicht. In der zugehörigen PDF-Druckquittung aber schon. *Lösung: minimale und maximale Zeichenzahl für diese Felder festlegen.*
 - **RTF-Texte mit speziellen Formatierungen** (hängende Einzüge, Hochstellungen, Aufzählungen, Zeilenausrichtung). Die Texte werden vollständig soweit möglich sinnvoll formatiert ausgegeben. *Eine Nachbearbeitung kann aber nötig sein.*
Hinweis: Seit der Version 6.10.1 können Texte und RTF-Texte auch direkt mit HTML-Formatierungen gestaltet werden (Eigenschaft „Sonderzeichen nicht nach HTML codieren“), z.B. mit , • oder <sup>.
Hinweis: Seit der Version 6.10.1 können alle Objekte wahlweise nicht nach HTML oder PDF ausgegeben werden. Dadurch können für die HTML-Ausfüllmaske und für die PDF-Druckquittung sehr einfach unterschiedliche Objekte verwendet werden die im Formulareditor übereinander liegen.
 - **Pflichtfeldprüfung im IE8** und älter. Hier wird u.a. das neue zusätzliche Attribut „required“ nicht korrekt erkannt. Bei allen Browsern mit solchen Problemen erscheint zu Beginn eine Meldung (kann inhaltlich angepasst und mit der JavaScript Funktion „OnBrowserIncompatible“ auch durch einen eigenen individuellen Text ersetzt werden, Siehe Abschnitt F.b.)


C) HTML-Layout und JavaScripte überprüfen

- Über das Menü „Datei“ -> „Exportieren als HTML“ oder den Button  das FVL-Formular nach HTML ausgeben lassen. Das HTML-Formular öffnet sich in Ihrem Standard-Browser. Der Code wird standardmäßig im selben Verzeichnis wie das FVL erzeugt.
Hinweis: Wenn Sie ein FVL in der Testphase immer wieder nach HTML exportieren wollen, so können Sie unter „Einstellungen für den Export nach HTML ...“ -> „Allgemein“ einstellen, dass die Fragen nach dem Überschreiben entfällt.
- Zielfeld ohne Rückfrage überschreiben, wenn diese bereits existiert
- Prüfen Sie im Browser das Layout: Ob Texte abgeschnitten sind (*Objekt-Rahmen in Editor vergrößern*), ob Elemente fehlen (*liegen evtl. im Hintergrund oder sind nicht HTML geeignet*) oder ob Elemente überlappen (*Objekte verschieben oder Objektgrößen anpassen, bei RTF evtl. auf Zeilenschaltung wechseln – siehe oben*)
 - **Hinweis:** Prüfen Sie evtl. auch in anderen Browsern (HTML-Datei mittels „Öffnen mit ...“ dort anzeigen lassen) soweit auf Ihrem Rechner vorhanden. Oder die „Browser-Optionen“ im Menü „Extras“ so einstellen, dass Sie nach dem Export gefragt werden, welcher Browser verwendet werden soll.
 - Testen Sie die Java-Script-Plausibilitäten. Bei Fehlern/nicht funktionierenden JS entweder im Browser per „F12“ (IE / FF /Chrome) oder über das Browser-Menü (ältere FireFox Versionen) den Debugger/die Fehlerkonsole einschalten. Dort werden die JavaScript Fehler angezeigt. Korrekturen dann im Formular-Editor bei den HTML-JavaScripten machen. Falls der Fehler unklar ist, mit „alert()“ Zwischenergebnisse ausgeben lassen.

Hinweis: Das Zwischenspeichern und das Absenden funktioniert normalerweise erst, wenn das Formular auf einen Formular-Server hochgeladen wurde.

- **Tipp:** Mit der Tastenkombination „Strg“ + „Alt“ + „h“ kann man direkt den Editor für die globalen HTML-JavaScripte öffnen
- Den Punkt C) solange durchlaufen, bis Sie mit dem Resultat zufrieden sind.
Ernste Probleme bitte mit dem FVL und einer Erläuterung per Mail an [bol](mailto:info@bol-systemhaus.de) senden
info@bol-systemhaus.de

D) Auf den Formular-Server hochladen

- Das FVL nun als ZIP-Datei exportieren. Button  oder Menü „Datei“.
Dabei wird das HTML mit allen Bildern und JavaScript sowie das PDF in einer ZIP-Datei erzeugt.
Hinweis: Eine passende ZIP-Datei für den Formular-Server muss als **Formularname.html.zip** benannt sein. Der Editor schlägt den passenden Namen automatisch vor.
Hinweis: Wenn bei den Bildern die Eigenschaft „Original-Namen verwenden“ angekreuzt wurde, so müssen die Original-Bilder per Hand in die ZIP-Datei kopiert werden.
- Diese ZIP-Datei – genau wie bisher die PDF-Datei – als neues (HTML-) Formular oder als neue Version eines (PDF-) Formulars auf den Formular-Server hochladen.
Hinweis: Die ZIP-Datei wird von Formular-Server automatisch ausgepackt und (ab Version 3.6 des FS) das enthaltene PDF automatisch als Druckquittung eingetragen (*SPATH*Formularname.pdf)
- Bei den ersten neuen HTML-Formularen eines Mandanten bitte kontrollieren, ob in HTML-Quellcode nach dem Instanzieren alle xxx...xxx Ersetzungsplatzhalter korrekt ersetzt wurden. Außerdem das Zwischenspeichern und das Einreichen testen. Bei Problemen die Ersetzungen am Formular-Server ändern (lassen).
- Bei neuer Version für ein bestehendes PDF-Formular ist das das HTML-Formular automatisch online, da es über den identischen findform-Link erreichbar ist.
Bei neuen HTML-Formularen diese im Web-Auftritt verlinken lassen (wie PDFs).

E) Besonderheiten für HTML-Formulare

- Für den PDF-Export werden die beiden Feld-Eigenschaften „**nicht veränderbar**“ und „**nicht betretbar**“ beide in ein schreibgeschütztes Feld umgewandelt. Für den HTML-Export jedoch wird „nicht veränderbar“ in „readonly“ und „nicht betretbar“ in „disabled“ umgewandelt.
Wichtig: In HTML werden Felder mit der Eigenschaft „disabled“ nicht mit abgesendet!!
Daher sollten schreibgeschützte Felder die per Script berechnet werden immer nur „readonly“ sein.
Wichtig: Ankreuzfelder kennen die Eigenschaft „readonly“ nicht, diese sollten daher „disabled“ sein.
- Beim Umwandeln der PDF-JavaScripte in HTML-JavaScripte werden einige Textersetzungen gemacht, um HTML-Besonderheiten zu berücksichtigen.
Die Ersetzungen findet man in der Datei „editor.scriptreplace“ im Installationsverzeichnis. Diese Datei kann u.U. auch für individuelle Textersetzungen genutzt werden.
- Wenn es im FVL das Feld „Form.FormPublish#2.LocalSaveURL#4“ (für die Zieladresse beim Zwischenspeichern) gibt, dann wird beim HTML-Export automatisch das unsichtbare Feld „Form.FormPublish#2.FindformURL#6“ mit den Standardwert „xxxFindformURLxxx“ erzeugt. Dieses Feld wird beim Zwischenspeichern von HTML-Formularen benötigt, um dort den Link auf die aktuelle Version des HTML-Formulars mitzutransportieren.
- Außerdem wird beim HTML Export in mehrere unsichtbare Felder ein Wert „xxx(Feldname)xxx“ eingetragen, um bei den Textersetzungen für HTML-Dateien von Formular-Server die richtigen Werte eintragen zu lassen.
- Wenn ein HTML-Formular (oder eine ZIP-Datei) am Formular Server mit der Technik „PDF“ hochgeladen wird (um als neue Version eines bestehenden PDF-Formulars zu fungieren), so wird durch den FS eine Ersetzung in den entsprechenden Formular-Feldern gemacht (wie bei PDF) unabhängig von Vorgabewerten in den Feldern.
Wenn es aber mit der Technik „HTML“ hochgeladen wird, so wird – wie bisher – eine Textersetzung im HTML-Code vorgenommen. D.h. in den Feldern müssen die entsprechenden xxx-Werte als Vorgabewerte enthalten sein.
- „Wasserzeichen“ (feste Vorlagen für einen Teil einer Formularseite) werden noch nicht mit nach HTML ausgegeben.
- Berechnete Gleitkommazahlen in HTML werden nicht automatisch formatiert. Hierfür muss für das berechnete Feld die Funktion „.formatValue()“ aufgerufen werden. Siehe Absatz F)a.
- Der Datumspicker funktioniert auch schon in IE8.
Für schreibgeschützte Felder kann er nicht (mehr) aufgerufen werden.
- Wenn normale Text-Objekte einen Objekt-Namen haben, so wird dieser auch mit nach HTML exportiert (sowohl als Attribut „name“, als auch als Attribut „id“, damit es auch in älteren IE-Versionen funktioniert).
Auf diese Weise kann per JavaScript (document.getElementsByName("name")[0]) auch auf Texte zugegriffen werden, um dieses z.B. un-/sichtbar zu machen oder farblich zu markieren.
- Tabs in RTF-Texten werden automatisch in einzelne Leerzeichen umgewandelt, um ein halbwegs sinnvolles Layout zu erzielen.
- Die „Umgebungsfarbe“ im HTML kann nur über die Datei „eforms.css“ im Installationsverzeichnis oder bei Verwendung eines eigenen CSS angepasst werden.
- Das erzeugte HTML funktioniert auch auf SmartPhone (iPhone, Android) und Tablet PCs.
- Im IE 9 wird bei HTML vom eigenen Rechner manchmal der „Kompatibilitätsmodus“ (d.h. das Verhalten wie IE8) verwendet. Dies kann per F12 oder im Menü „Extras“ ausgeschaltet werden.

F) Beispiele für nützliche HTML-JavaScripte

a. Formatieren von berechneten Zahlenfeldern

Zahlenfelder werden beim „Verlassen“ automatisch formatiert. Berechnete Felder sollten „readonly“ sein und nach der Berechnung per JS formatiert werden (z.B. Hinzufügen von Nachkomastellen):

```
var f = this.getField("test");
...
f.value = result;
f.formatValue();
```

b. Funktion zur Prüfung auf IE 8 oder Kompatibilitätsmodus (in den globalen HTML-JavaScripten)

Einige in HTML verwendete Hilfsfunktionen/Eigenschaften funktionieren leider nicht im IE 8 oder im sogenannten „Kompatibilitätsmodus“ (mit dem der IE 8 simuliert wird). Normalerweise erscheint dann beim Öffnen des Formulars eine Standard-Warnung. Mit der Funktion „**OnBrowserIncompatible**“ in den globalen HTML-JavaScripten kann man alternativ selbst einen Text ausgeben und evtl. weitere JavaScript-Aktionen ausführen:

```
function OnBrowserIncompatible (s) {
    // anderen oder besseren Hinweis als den Standardtext ausgeben
    alert("Sie verwenden leider den Internet Explorer 8 (oder älter) bzw. haben die
    'Kompatibilitätsansicht' im Internet Explorer aktiviert.\r\rDadurch kann dieses Formular nicht
    geprüft und abgesendet werden.\r\r Bitte deaktivieren Sie entweder die Kompatibilitätsansicht
    unter 'Extra'->'Einstellungen zur Kompatibilitätsansicht' oder verwenden Sie bitte einen anderen
    Browser.");
    //optional den oder die Absende-Buttons (mit Namen „Absenden“) ausblenden
    var a = document.getElementsByName("Absenden");
    var l = a.length;
    for (var i = 0; i < l; i++) {
        a[i].style.visibility = "hidden";
    }
}
```

c. Druck-Button im HTML für direkten PDF-Druck

Das normale „window.print();“ führt dazu, dass das HTML-Dokument an den Drucker gesendet wird. Dies führt oft zu unschönen Seitenumbrüchen sowie zu meist unerwünschten Angaben oben und unten auf den Ausdruckseiten.

Daher kann ein Druck des zugehörigen PDFs mit Daten wesentlich sinnvoller sein. Dies wird durch folgendes HTML-JavaScript für den Druck-Button erreicht:

```
var ziel = "https://formular-gateway.de/NetGateway/DisplayPDF?action=getnowpdfa";
// diese URL an Ihre Installation anpassen!
var form = document.getElementsByName("WaimeaForm")[0];
form.target = '_blank'; // Neues Seite öffnen - HTML bleibt geöffnet erhalten
var a = form.action; // Alte Ziel-URL speichern
form.action = ziel; // Neue Ziel URL setzen
OnSubmit (true); // „true“ bedeutet: Pflichtfeldprüfung überspringen
form.submit(); // PDF aufrufen
form.target = '_self'; // Auf alten Wert zurücksetzen
form.action = a; // Auf alten Wert zurücksetzen
```

d. Script für das Zwischenspeichern

Für das Zwischenspeichern müssen nicht mehr – wie in PDF – alle Felder auf „nicht-Pflichtfeld“ gesetzt werden. Es genügt beim Absenden mit submitForm als zweiten Parameter „true“ zu verwenden. Damit werden alle Prüfungen auf Vollständigkeit unterdrückt.

```
var ziel = this.getField("Form.FormPublish#2.LocalSaveURL#4").value;
this.submitForm( ziel, true );
```

e. Text zu Ankreuzfeld rot machen

In mehreren Browsers ist es nicht möglich die Hintergrundfarbe von Ankreuzfeldern (z.B. bei der Vollständigkeitsprüfung) farblich zu verändern. Um den Nutzer trotzdem direkt auf fehlende Ankreuzfelder hinzuweisen, könnte es Sinn machen, den zugehörigen Text rot zu machen. Dazu muss dieser Text in eforms einen eindeutigen „Namen“ erhalten. Im folgenden Beispiel

- Ja, ich habe die Hinweise zum Datenschutz zur Kenntnis genommen und akzeptiere die genannten Bedingungen.

muss der Text, der rot werden soll, genauso heißen wie das Ankreuzfeld mit der Erweiterung „_text“ am Ende. Diese Bezeichnung muss bei dem Text als „Name“ eingetragen sein. Z.B. Ankreuzfeld „Datenschutz“ und der Text (kein Label verwenden!) heißt dann „Datenschutz_text“:

```
if (feld.type == "checkbox") {
    if (feld.required && !feld.checked) {
        if (fanzahl == 0) firsterrorfield = feld;
        feld.fillColor = color.red;
        document.getElementsByName(feld.name+"_text")[0].style.color = color.red;
        fanzahl++;
        if (feld.userName == "")
            fstring += "Das Feld " + feld.name + " ist ein Pflichtfeld.\n";
        else fstring += "Das Feld " + feld.userName + " ist ein Pflichtfeld.\n";
    }
    else {
        feld.fillColor = color.white;
        document.getElementsByName(feld.name+"_text")[0].style.color = color.black;
    }
}
```

f. Berechnungen HTML

In HTML wird mit „.value“ immer ein String zurückgeliefert. Zum Rechnen mit Zahlenfeldern muss daher „.valueAsInt“ bzw. „.valueAsFloat“ verwendet werden. Folgende Berechnungen mit Menge (Ganze Zahl) und Preis (Gleitkommazahl) können für HTML als Vorlage dienen:

```
if (this.getField("Menge").value != "" && this.getField("Preis").value != "") {
    this.getField("Gesamt").value = this.getField("Menge").valueAsInt * this.getField("Preis").valueAsFloat;
    this.getField("Gesamt").formatValue();
    summe = summe + this.getField("Menge").valueAsInt * this.getField("Preis").valueAsFloat;
} else this.getField("Gesamt").value = "";
```

g. Auswahlfelder befüllen

Anders als in PDF kann man Auswahlfeldern (Pulldown) in HTML nicht einfach einen Array von Arrays als Text/Werte-Paare zuweisen. Die folgende Hilfsfunktion „populateSelectBox“ kann stattdessen verwendet werden (z.B. abspeichern der Funktion in den globalen HTML-JS)

```
function populateSelectBox(selectBox, sourceArray) {
    // Das Auswahlfeld 'selectBox' leeren und dann mit Daten aus Array-Array-Feld 'sourceArray' befüllen
    // Das Feld leeren
    selectBox.length = 0;
    // Array durchlaufen
    for (var i=0; i<sourceArray.length; ++i) {
        var thisResult = sourceArray[i];
        // Neuen Select-Eintrag erzeugen
        var tagNewOption = document.createElement('option');
        selectBox.appendChild(tagNewOption);
        // Wert aus dem Unter-Array verwenden
        tagNewOption.setAttribute('id', thisResult[1]);
        tagNewOption.setAttribute('value', thisResult[1]);
        // Text aus dem Unter-Array verwenden
        var optionText = document.createTextNode(thisResult[0]);
        tagNewOption.appendChild(optionText);
    }
}
```

Aufruf für ein Auswahlfeld „feld“ dann – wie bisher in PDF – z.B. per
populateSelectBox(feld,[["Frau", "w"],["Herr", "m"],["Firma", "f"]]);

h. Abhängige Auswahlfelder

Wenn ein zweites Auswahlfeld in Abhängigkeit von einem ersten Auswahlfeld befüllt werden soll, so sollte das folgende Script sowohl unter „Verlassen des Objekts“ als auch unter „Klicken auf das Objekt“ beim ersten Auswahlfeld eingehängt werden.

```
var a = this.getField("auswahlfeld1").value;
var s = this.getField("auswahlfeld2");

switch (a) {
  case "0":
    populateSelectBox(s,["Bitte zuerst den Hochschulort wählen!", "0"]);
    break;

    case "1":
    populateSelectBox(s,[
      ["TH Aachen", "1480"],
      ["H für Musik Köln, Abt. Aachen", "2582"],
      ["FH Aachen, Abt. Aachen", "5711"],
      ["Kath. Hochschule Nordrhein-Westfalen in Aachen (FH)", "6032"],
      ["Priv. FH für Ökonomie und Management Essen in Aachen", "810B"]]);
    break;

    case "2":
    populateSelectBox(s,[
      ["H für Technik Aalen (FH)", "6710"]]);
    break;

    ...
}
```

i. Zählen von verbleibenden Zeichen für mehrzeilige Textfelder

In HTML kann man für mehrzeilige Felder zwar eine maximale Zeichenzahl festlegen, aber – anders als in PDF – nicht verhindern, dass das Feld durch zu viele Zeilenumbrüche überläuft. was in HTML aber üblich ist – ohne dass dadurch das Problem wirklich behoben wird – ist ein Zähler, der die noch verbleibenden Zeichen anzeigt. Beispiel:

Sie können noch 35 Zeichen eingeben

Dies ist mit dem HTML-Event „Beim Loslassen einer Taste“ (Event onkeyup) möglich. Die Funktion „count“ stellt man in die globalen HTML-JavaScripte

```
function count(max, t, z) {
  if (t.value.length < max + 1)
    z.value = max - t.value.length;
  else {
    alert("Maximale Zeichenzahl erreicht!");
    t.value = t.value.substring(0, max);
    z.value = 0;
  }
}
```

Bei dem mehrzeiligen Textfeld „text1“ muss dann der Aufruf unter „Loslassen einer taste“ eingetragen werden. Das Zähler-Feld soll hier „test1_z“ heißen (und ist schreibgeschützt!), die Grenze liegt bei 40:

```
count(40, this.getField("text1"), this.getField("test1_z"));
```